SOLID PRINCIPI

- **Single Responsibility Principle** – Svaka klasa treba imati samo jednu ulogu

Ovaj princip je ispoštovan, jer svaka klasa radi ono za šta je i namijenjena. Jedina koja bi mogla potencijalno narušiti ovaj princip je klasa EParking (ona vrši npr. plaćanje).

- **Open/Closed Principle** – Klasa treba biti otvorena za nadogradnje, ali zatvorena za modifikacije

Klase koje bi potencijalno mogle narušavati ovaj princip su klase EParking, Vlasnik, Korisnik, Transakcija i Zahtjev, jer klasa EParking ima kao atribute druge klase (Gost, Clan, Administrator, Vlasnik, ParkingLokacija, Transakcija), klasa Transakcija sadrži atribute tipa ParkingLokacija i Korisnik, klasa Vlasnik ima atribute tipa Zahtjev i ParkingLokacija, klasa Zahtjev ima atribute tipa Korisnik i Vozilo, a klasa Korisnik ima atribut tipa Vozilo. Međutim, kako ove klase ne vrše modifikaciju navedenih atributa, već ih samo koriste kao atribute (ili eventualno u listi), ovaj princip je ispoštovan.

- **Liskov Substitution Principle** – Svaka osnovna klasa treba biti zamjenjiva svim svojim podtipovima bez da to utječe na ispravnost rada programa

U našem slučaju je ovaj princip zadovoljen, jer na svim mjestima gdje koristimo baznu klasu Korisnik možemo koristiti i izvedene klase Gost i Clan.

- **Interface Segregation Principle** – Bolje je imati više specifičnih interfejsa, nego jedan generalizovani

Ovaj princip je zadovoljen, jer interface-i IGoogleMaps, IPayPal i ICreditCard obavljaju samo jednu vrstu akcija; IGoogleMaps obavlja funkciju računanja rute, IPayPal obavlja funkciju plaćanja pomoću PayPal računa, a ICreditCard obavlja funkciju plaćanja pomoću kartice.

- **Dependency Inversion Principle –** Sistem klasa i njihovo funkcionisanje treba ovisiti o apstrakcijama, a ne o konkretnim implementacijama

Ovaj princip je ispoštovan, jer je bazna klasa Korisnik ujedno i apstraktna klasa iz koje su izvedene klase Gost i Clan.